Deployment Manager Fundamentals

The following components are the fundamentals of Deployment Manager.

Configuration

A configuration describes all the resources you want for a single deployment. A configuration is a file written in YAML syntax that lists each of the resources you want to create and its respective resource properties. A configuration must contain a `resources:` section followed by the list of resources to create.

Each resource must contain three components:

- `name` - A user-defined string to identify this resource such as `my-vm`, `project-data-disk`, `the-test-network`.
- `type` - The type of the resource being deployed such as `compute.v1.instance`, `compute.v1.disk`. The base resource types are described and listed on the Supported Resource Types (/deployment-manager/docs/configuration/supported-resource-types) documentation.
- `properties` - The parameters for this resource type. They must match the properties for the type such as `zone: asia-east1-a`, `boot: true`.

This is an example configuration:

```
rces:
e: the-first-vm
e: compute.v1.instance
perties:
one: us-central1-a
achineType: https://www.googleapis.com/compute/v1/projects/myproject/zones/us-centra
isks:
 deviceName: boot
 type: PERSISTENT
 boot: true
 autoDelete: true
 initializeParams:
   sourceImage: https://www.googleapis.com/compute/v1/projects/debian-cloud/global/i
```

```
etworkInterfaces:
 network: https://www.googleapis.com/compute/v1/projects/myproject/global/networks/d
 accessConfigs:
 - name: External NAT
   type: ONE_TO_ONE_NAT
```

Templates

A configuration can contain templates, which are essentially parts of the configuration file that has been abstracted into individual building blocks. After you create a template, you can reuse them across deployments as necessary. Similarly, if you find yourself rewriting configurations that share very similar properties, you can abstract the shared parts into templates. Templates are much more flexible than individual configuration files and intended to support easy portability across deployments.

A template file is written in either Python or Jinja2. The Deployment Manager system will interpret each template recursively and inline the results within the configuration file. As such, the interpretation of each template eventually results in the same YAML syntax for resources as that defined above for the configuration file itself.

To create a simple template, read Creating a Basic Template (/deployment-manager/docs/configuration/templates/create-basic-template).

Configurations are described as fully-expanded or unexpanded. A fully-expanded configuration describes all resources and properties of the deployment, including any content from imported template files. For example, you would supply an unexpanded configuration that uses a template like so:

```
ts:
h: vm_template.jinja

rces:
e: vm-instance
e: vm_template.jinja
perties:
one: us-central1-a
roject: myproject
```

Once expanded, your configuration file would contain the contents of all your templates, like so:

```
rces:
e: the-first-vm
e: compute.v1.instance
perties:
one: us-central1-a
achineType: https://www.googleapis.com/compute/v1/projects/myproject/zones/us-centra
isks:
 deviceName: boot
 type: PERSISTENT
 boot: true
 autoDelete: true
 initializeParams:
   sourceImage: https://www.googleapis.com/compute/v1/projects/debian-cloud/global/i
   networkInterfaces:
   - network: https://www.googleapis.com/compute/v1/projects/myproject/global/networ
ccessConfigs:
 name: External NAT
 type: ONE_TO_ONE_NAT
```

Resource

A resource represents a single API resource. This can be an API resource provided by a Google-managed base type or an API resource provided by a Type Provider. For example, a Compute Engine instance is a single resource, a Cloud SQL instance is a single resource, and so on.

To specify a resource, you provide a **Type** for that resource. See the Types section below to learn more about types.

Types

To create a resource in Deployment Manager, you must specify a `type`. A type can represent a single API resource, known as a base type, or a set of resources, known as a composite type, that will be created as part of your deployment.

For example, to create a Compute Engine VM instance, specify the corresponding base type like so in your configuration:

```
rces:
e: the-first-vm
e: compute.v1.instance # The type of resource to deploy
```

Deployment Manager offers a list of base types maintained by Google that you can use immediately. You can find a list of these types in the Supported resource types and properties (/deployment-manager/docs/configuration/supported-resource-types) documentation.

Base types and type providers

A base type creates a single primitive resource. For example, Google-owned base types include `compute.v1.instance`, `storage.v1.bucket`, and `sqladmin.v1beta4.database`, all of which are served by the respective Compute Engine V1 API, Cloud Storage V1 API, and the Cloud SQL v1beta4 Admin API.

Base types are supported by a RESTful API that supports Create, Read, Update, and Delete (CRUD) operations. You can also create additional base types by adding a **type provider** if the Google-owned types alone do not meet your needs. Creating a type provider exposes all resources of an API as base types that you can use. To create a type provider, you must supply an API descriptor document, which can be an OpenAPI specification (https://github.com/OAI/OpenAPI-Specification) or a Google Discovery (https://developers.google.com/discovery/v1/reference/apis), adjust any necessary input mappings (/deployment-manager/docs/configuration/type-providers/advanced-configuration-options#specifying_input_mappings) for the API, and register the type with Deployment Manager. Once created, you and other users with access to your project can use the types provided by the provider.

When you add a type provider, all resources that are provided by the API and supported by a RESTful interface with Create, Read, Update, and Delete (CRUD) (/deployment-manager/docs/configuration/type-providers/api-requirements) operations will be exposed as types that you can use in your deployment.

Creating your own type provider is an advanced scenario, and Google recommends that you do this only if you are very familiar with the API you want to integrate.

To learn how to create a type provider, see Integrating with Deployment Manager (/deployment-manager/docs/configuration/type-providers/process-adding-api).

When you call a base type in your templates or configurations, you use one of the following syntaxes depending on the type.

- For Google-managed base types, use:

  ```
  type: [API].[VERSION].[RESOURCE]
  ```

  For example, `compute.v1.instance`.

- For Google-managed type providers (beta), use:

  ```
  type: gcp-types/[PROVIDER]:[RESOURCE]
  ```

  For a list of supported type providers, see Supported GCP type providers (/deployment-manager/docs/configuration/supported-gcp-types).

- For base types provided by a type provider, use:

  ```
  type: [PROJECT_ID]/[TYPE_PROVIDER]:[COLLECTION]
  ```

  Where `[COLLECTION]` is the path to the API resource to deploy.

Composite types

A composite type contains one or more templates (#templates) that are preconfigured to work together. These templates expand to a set of base types when deployed in a deployment. Composite types are essentially hosted templates that you can add to Deployment Manager. You can create composite types for common solutions so that the solution is easily reusable, or create complex setups that you can reuse in the future.

For example, you can create a composite type that deploys a network load balanced managed instance group. A network load balancer requires multiple Google Cloud Platform resources and some configuration between resources, so you can set up these resources in a configuration

once and register the type with Deployment Manager. Afterwards, you and other users with access to your project can call that type and deploy it in future configurations.

To call a composite type in your configuration, use:

```
[PROJECT_ID]/composite:[TYPE_NAME]
```

For example:

```
rces:
e: my-composite-type
e: myproject/composite:example-composite-type
```

To learn how to create a composite type, read Adding a Composite Type to Deployment Manager (/deployment-manager/docs/configuration/composite-types/creating-composite-types).

Manifest

A manifest is a read-only object that contains the original configuration you provided, including any imported templates, and also contains the fully-expanded resource list, created by Deployment Manager. Each time you update a deployment, Deployment Manager generates a new manifest file to reflect the new state of the deployment. When troubleshooting an issue with a deployment, it is useful to view the manifest.

For more information, see Viewing a Manifest (/deployment-manager/docs/deployments/viewing-manifest).

Deployment

A deployment is a collection of resources that are deployed and managed together, using a configuration.

For more information, see Creating a Deployment (/deployment-manager/docs/deployments).

What's next

- Go through the Deployment Manager quickstart (/deployment-manager/docs/quickstart).
- Walk through the Step-by-step Guide (/deployment-manager/docs/step-by-step-guide).
- Learn about configurations (/deployment-manager/docs/configuration) and deployments (/deployment-manager/docs/deployments).

Last updated 2020-06-22 UTC.